

FIG. 1

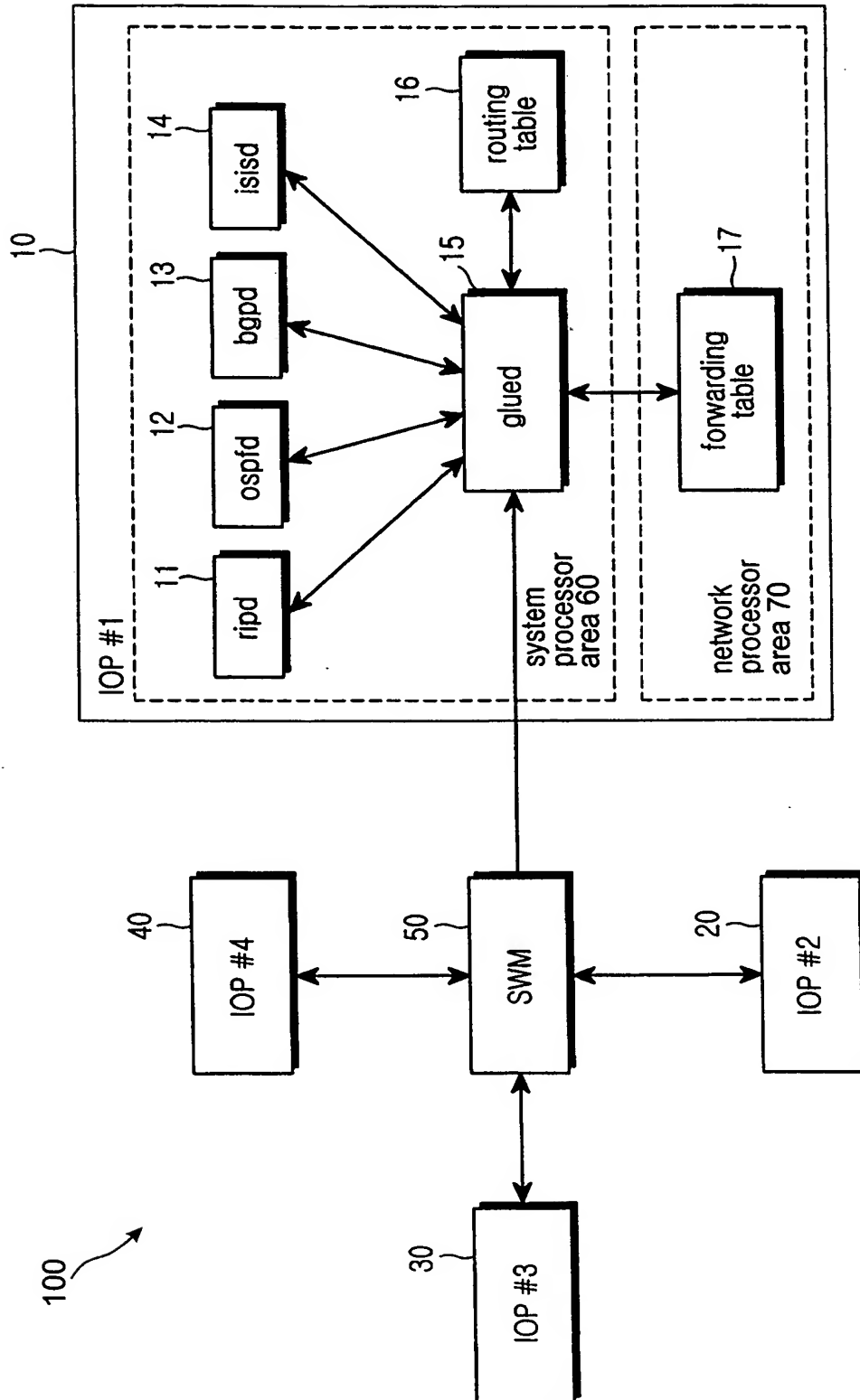


FIG. 2

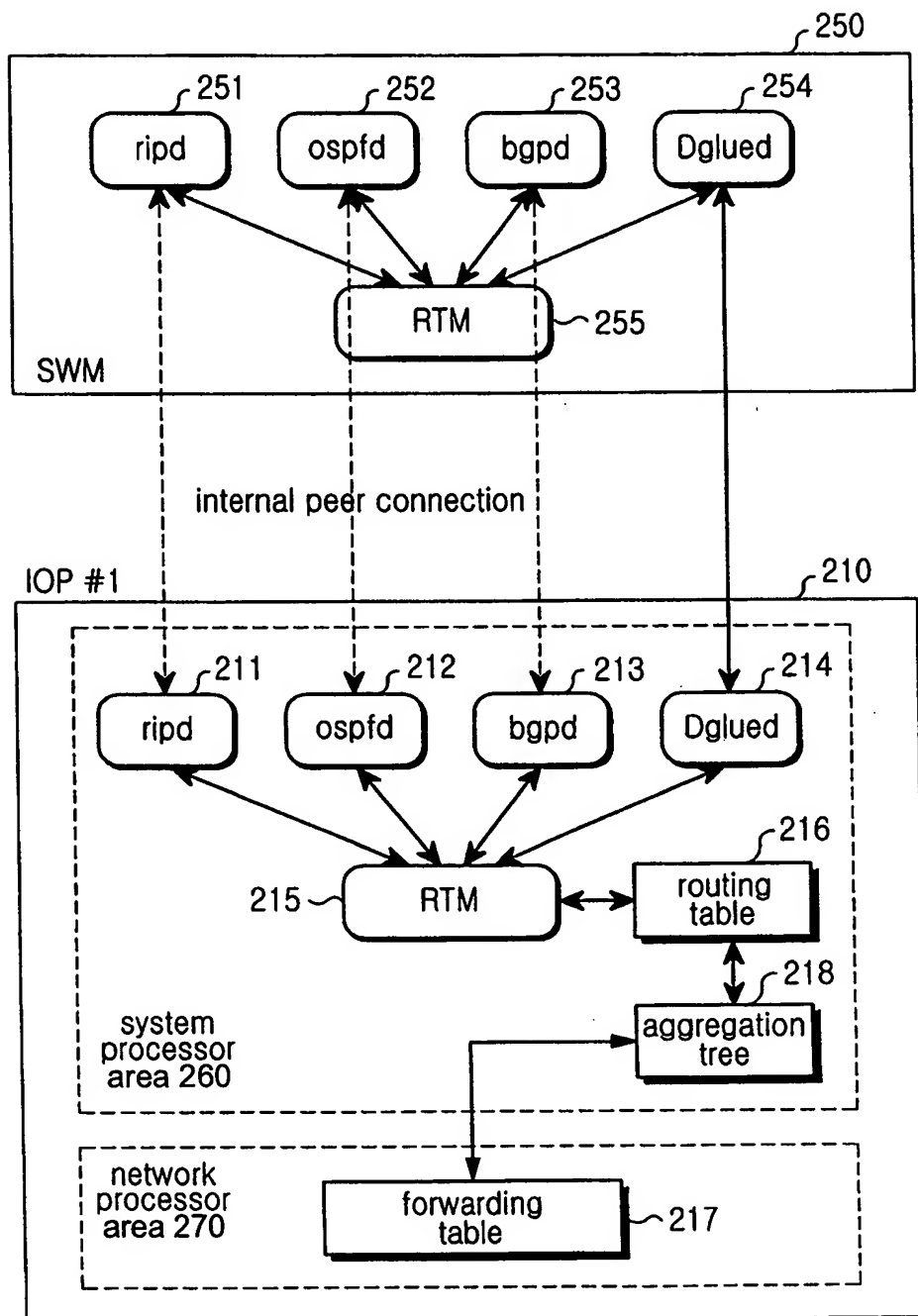


FIG. 3

PREFIX	LENGTH	AGGREGATION_ LEVEL	INDEX	TYPE	SOURCE IOP	FT FLAG	PARENT	L_SUBTREE	R_SUBTREE
--------	--------	-----------------------	-------	------	------------	---------	--------	-----------	-----------

FIG. 4A

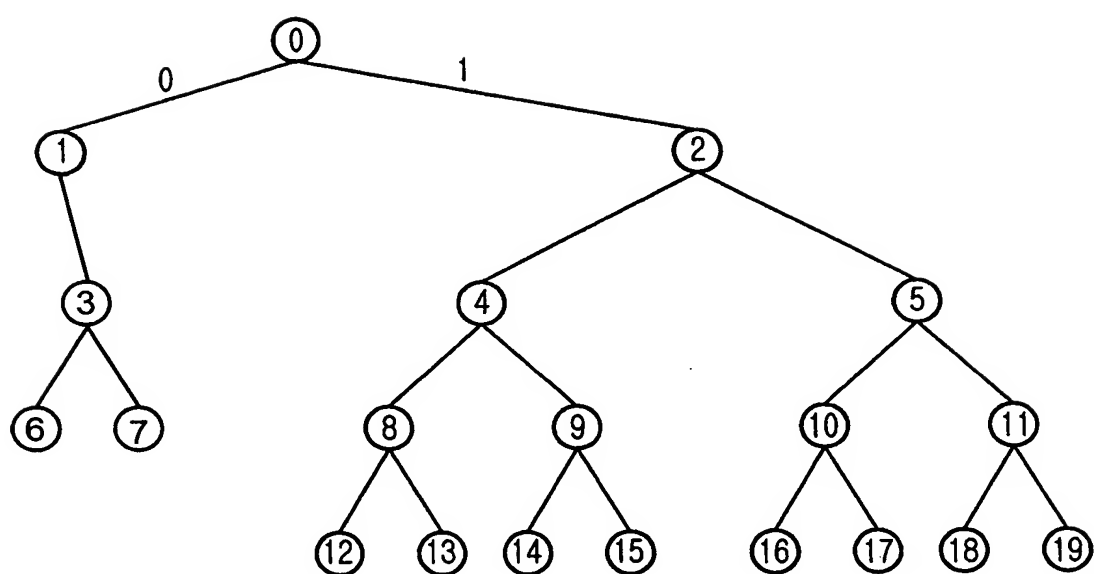


FIG. 4B

6/21

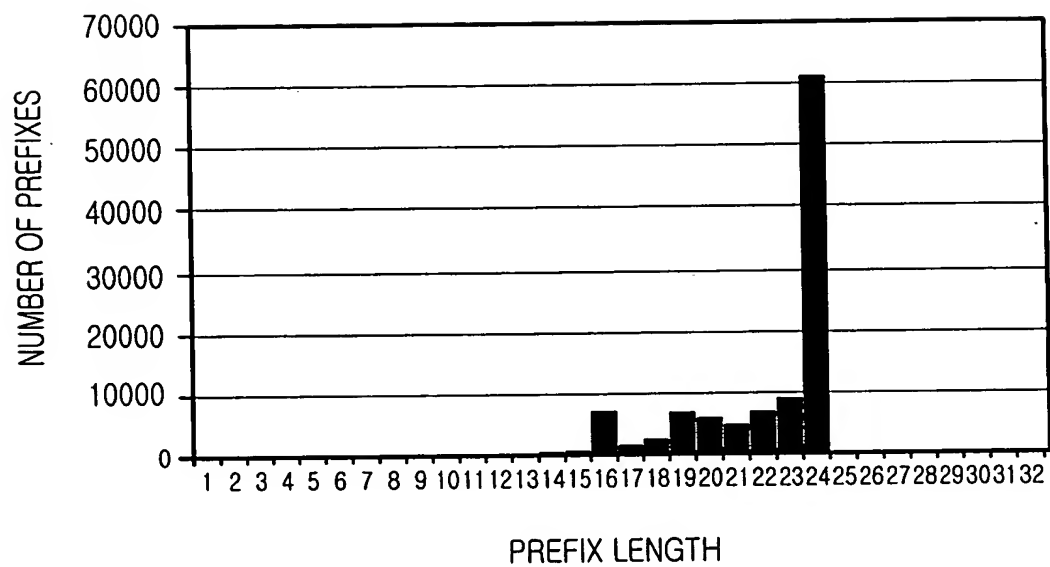


FIG.4C

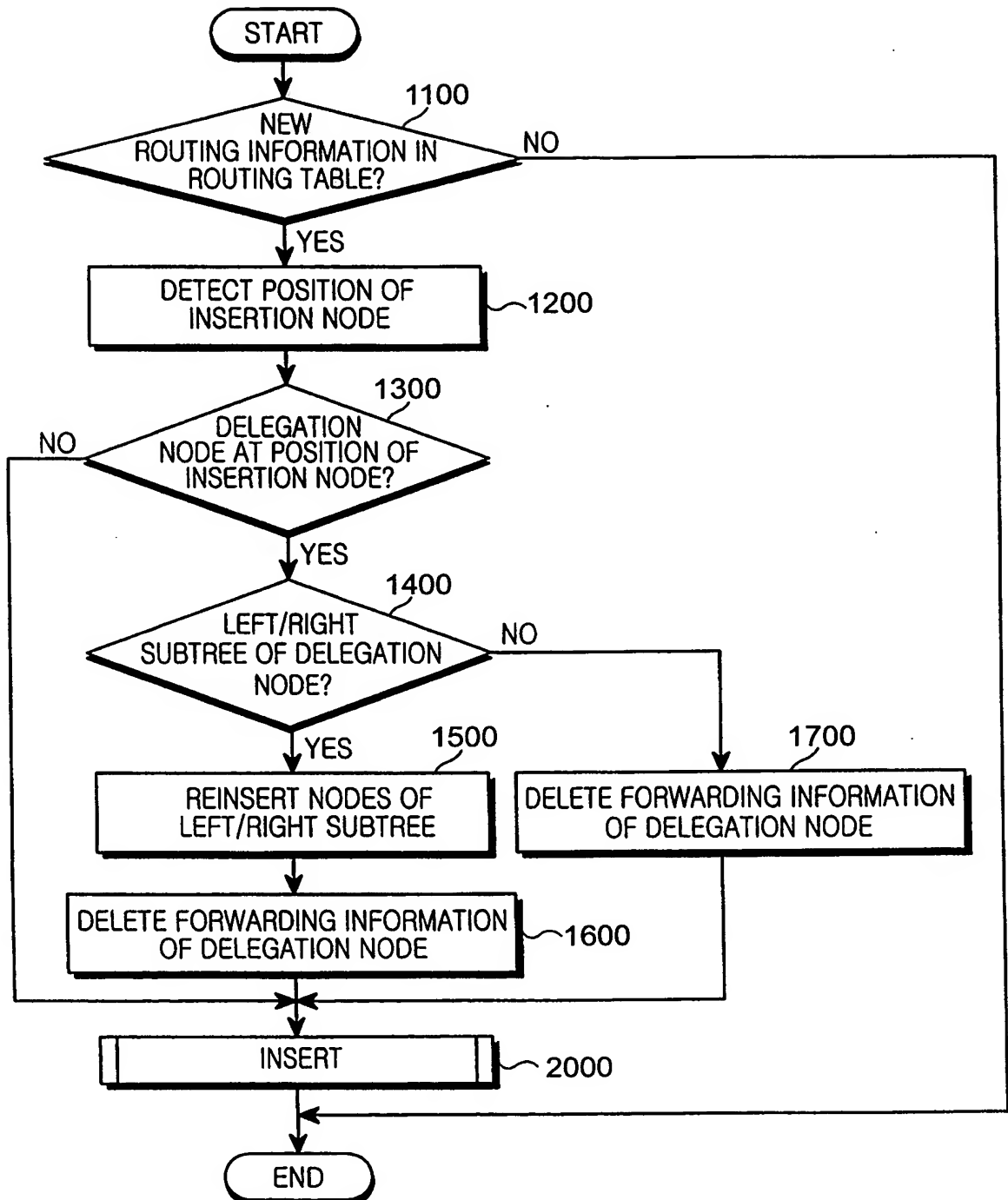


FIG. 5

FIG. 6A

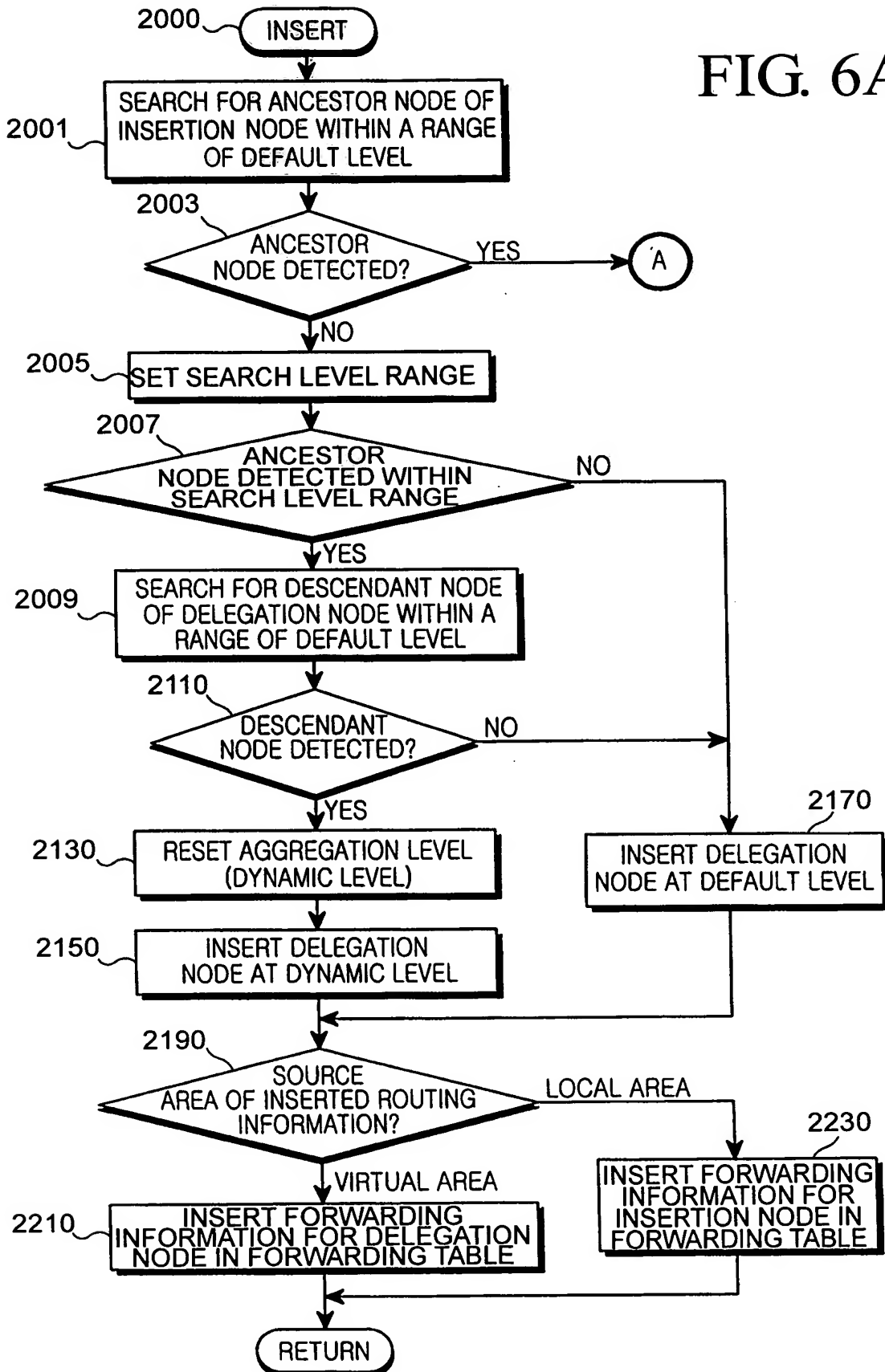
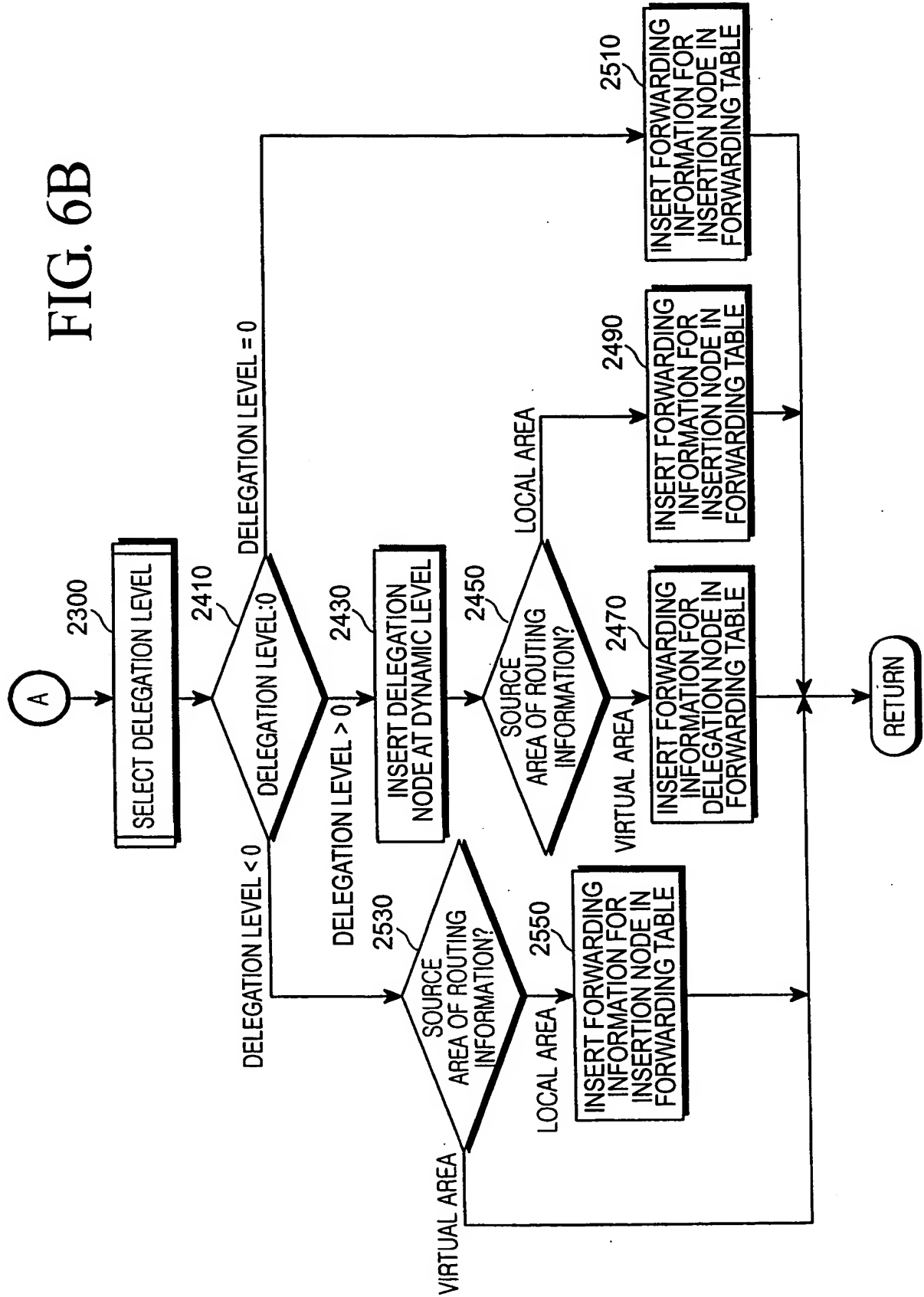


FIG. 6B



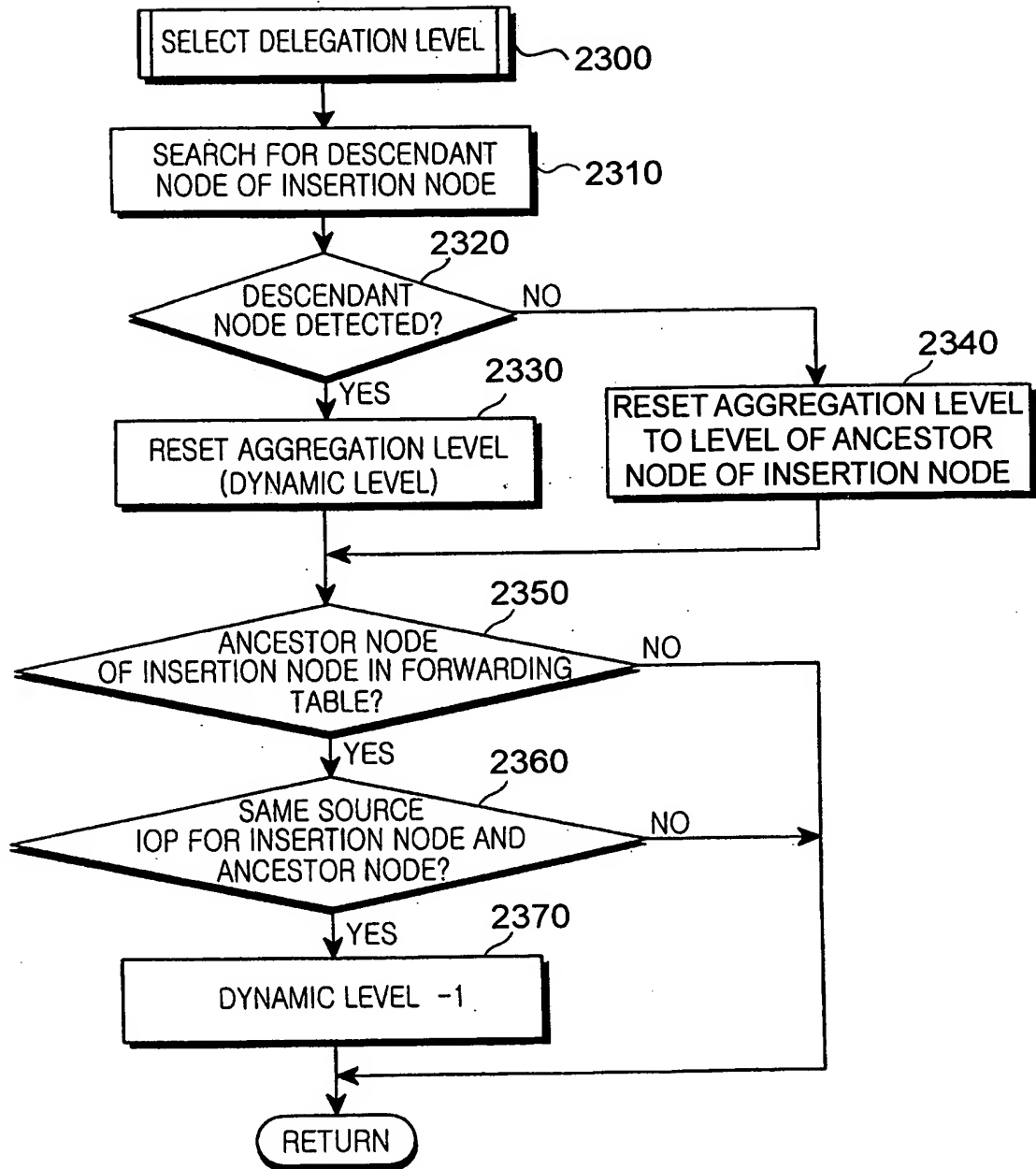
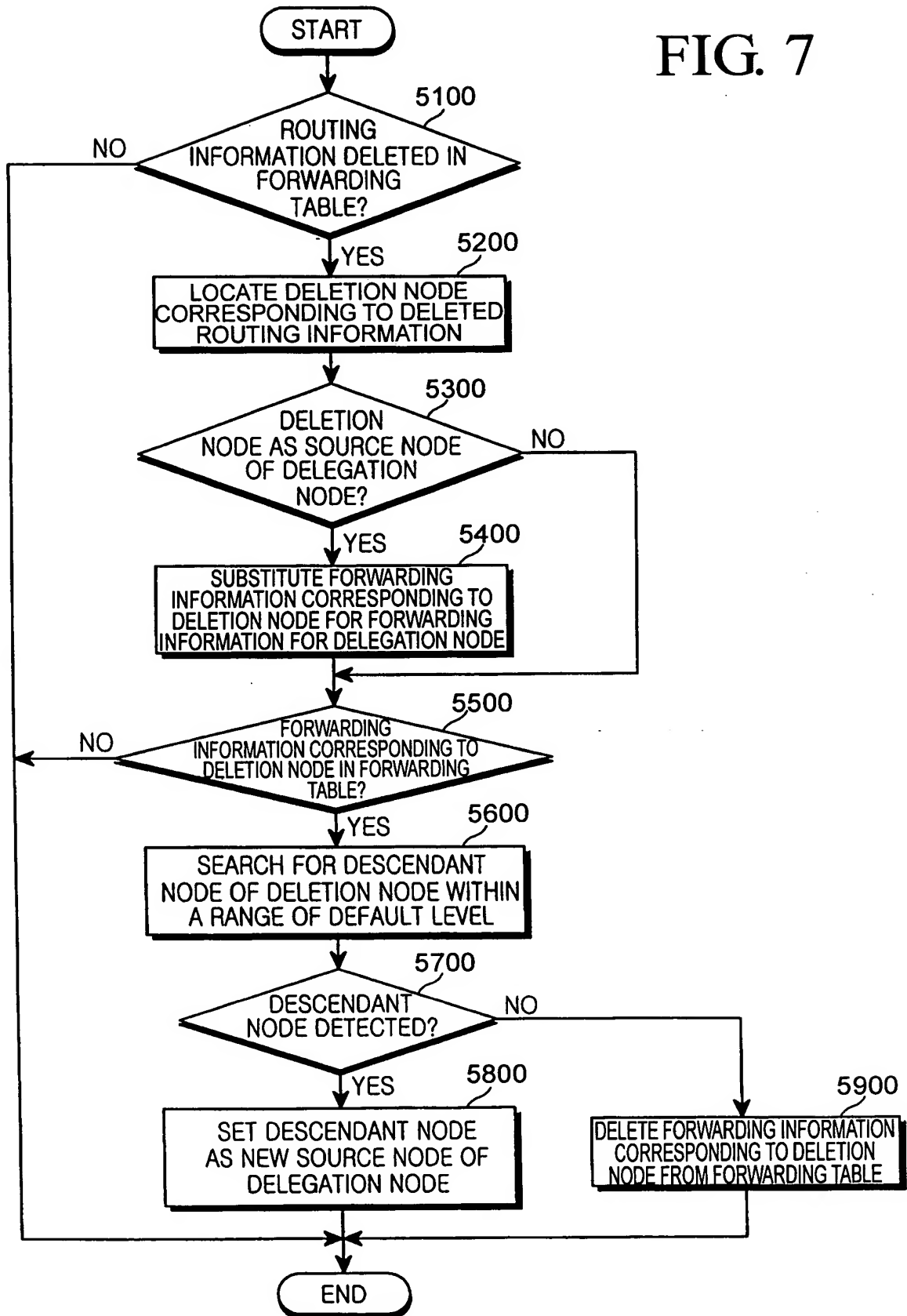


FIG. 6C

FIG. 7



```

Insertion (prefix) {
    local insertionnode, ancestornode, result
    /* STEP 1 */
    insertionnode := FindNode (prefix)
    /* STEP 2 */
    /* Virtual node exist in the aggregation tree */
    if (insertionnode ≠ NIL) then
        result := LeftSubTree (insertionnode)
        if (result ≠ NIL) then
            Insertion(result)
        result := RightSubTree (insertionnode)
        if (result ≠ NIL) then
            Insertion(result)
        DeleteNodeFromFT (insertionnode)
    /* STEP 3 */
    /* FindAncestorNode searches ancestor node */
    ancestornode := FindAncestorNode (insertionnode, defaultlevel)
    if (ancestornode ≠ NIL) then
        InsertWithAncestorNode (insertionnode, ancestornode)
    else
        InsertWithoutAncestorNode (insertionnode)
}

```

FIG.8A

```

FindNode (prefix) {
    local node
    /* Search for node to be inserted */
    node := GetNode (prefix)
    /* Identity the insertion node */
    if (node ? NIL) then
        return (node)
    else
        return NIL
}

```

FIG.8B

```

InsertWithoutAncestorNode (insertionnode) {
    local ancestornode, newdelegationnode, searchlevel,
        descendantnode, dynamiclevel
    /* Lemma 2-2 */
    searchlevel := defaultlevel*2 - 1
    ancestornode := FindAncestorNode (insertionnode, searchlevel)
    if (ancestornode = NIL) then
        dynamiclevel := defaultlevel
    else /* FindDescendantNode finds node not written to FT */
        descendantnode := FindDescendantNode (insertionnode, defaultlevel)
    /* Get adaptive level */
    if (descendantnode ? NIL) then
        dynamiclevel := GetNodePrefixLength (insertionnode)
            - GetDistPrefixLength (insertionnode, descendantnode)
    else
        dynamiclevel := defaultlevel
    newdelegationnode := MakeDelegationNode (insertionnode, dynamiclevel)
    if (NodeNextHopVirtual (insertionnode) = TRUE) then
        InsertNodeToFT (newdelegationnode)
    else /* All inter-domain node must be inserted into FT */
        InsertNodeToFT (insertionnode)
}

```

FIG.8C

```

MakeDelegationNode (node, level) {
    local delegationnode
    /* Make a virtual delegation node and set node type */
    delegationnode := AllocateDelegationNode (node, level)
    SetNodeType (delegationnode) := DELEGATION
    SetDelegationNodeIndex (delegationnode, node)
    return (delegationnode)
}

```

FIG.8D

```

InsertWithAncestorNode (insertionnode, ancestornode) {
    local newdelegationnode, result, dynamiclevel
    dynamiclevel := SelectDelegationLevel (insertionnode, ancestornode)
    if (dynamiclevel > 0) then
        newdelegationnode := MakeDelegationNode (insertionnode, dynamiclevel);
        /* Resolve destination area */
        if (NodeNexthopVirtual (insertionnode) = TRUE) then
            InsertNodeToFT (newdelegationnode)
        else
            InsertNodeToFT (insertionnode)
    elseif (dynamiclevel = 0) then
        InsertNodeToFT (insertionnode)
    else /* Check aggregatability of route */
        if (NodeNexthopVirtual (insertionnode) = FALSE) then
            InsertNodeToFT (insertionnode)
    }

```

FIG.8E

```

SelectDelegationLevel (insertionnode, ancestornode) {
    local dynamiclevel, result, searchnode, descendantnode
    descendantnode := FindDescendantNode (insertionnode, GetNodeLevel (ancestornode))
    /* Get adaptive level */
    if (descendantnode ≠ NIL) then
        dynamiclevel := GetNodePrefixLength (insertionnode)
                        - GetDistPrefixLength (insertionnode, descendantnode)
    else
        dynamiclevel := GetNodesLevel (insertionnode, ancestornode)
    /* Aggregatable case */
    if (CheckNodeUpdateToFT(ancestornode) = TRUE
        && NodeSource(insertionnode) = NodeSource(ancestornode)) then
        dynamiclevel := -1
    return dynamiclevel
}

```

FIG.8F

```

Deletion (prefix) {
  local deletionnode, delegationnode,
    result, descendantnode
  /* STEP1 */
  deletionnode := FindNode (prefix)
  /* STEP2 */
  if (CheckNodeAggregate (deletionnode) = TRUE)
    then
      deletionnode := GetDelegationNode (deletionnode)
  if (CheckNodeUpdateToFT (deletionnode) = TRUE)
    then
      descendantnode := FindDescendantNode
        (deletionnode, defaultlevel)
    /* Suppress deletion from FT */
    if (descendantnode ? NIL) then
      SetDelegationNode (deletionnode,
        descendantnode)
    else
      DeleteNodeFromFT (deletionnode)
}

```

FIG.8G

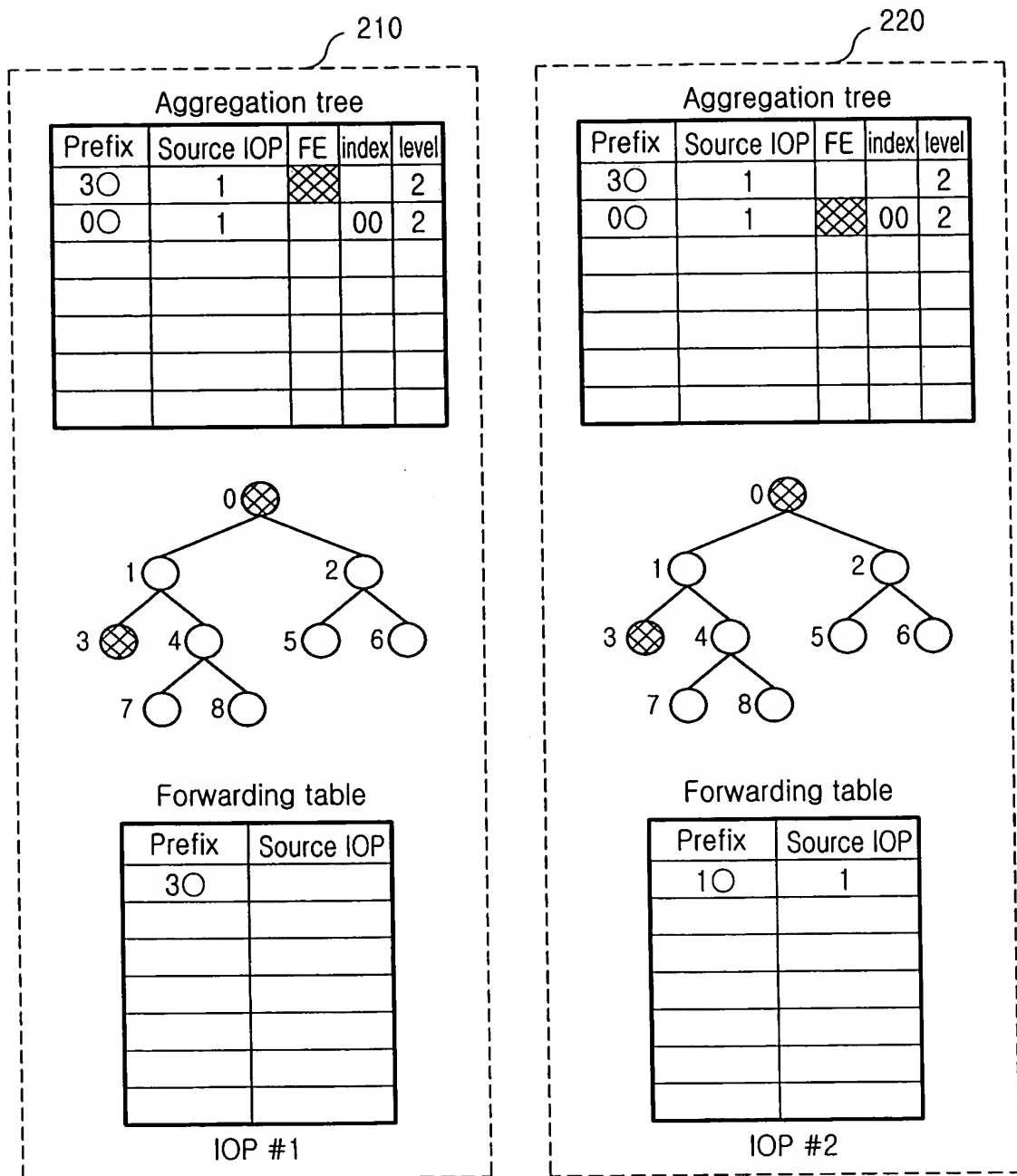
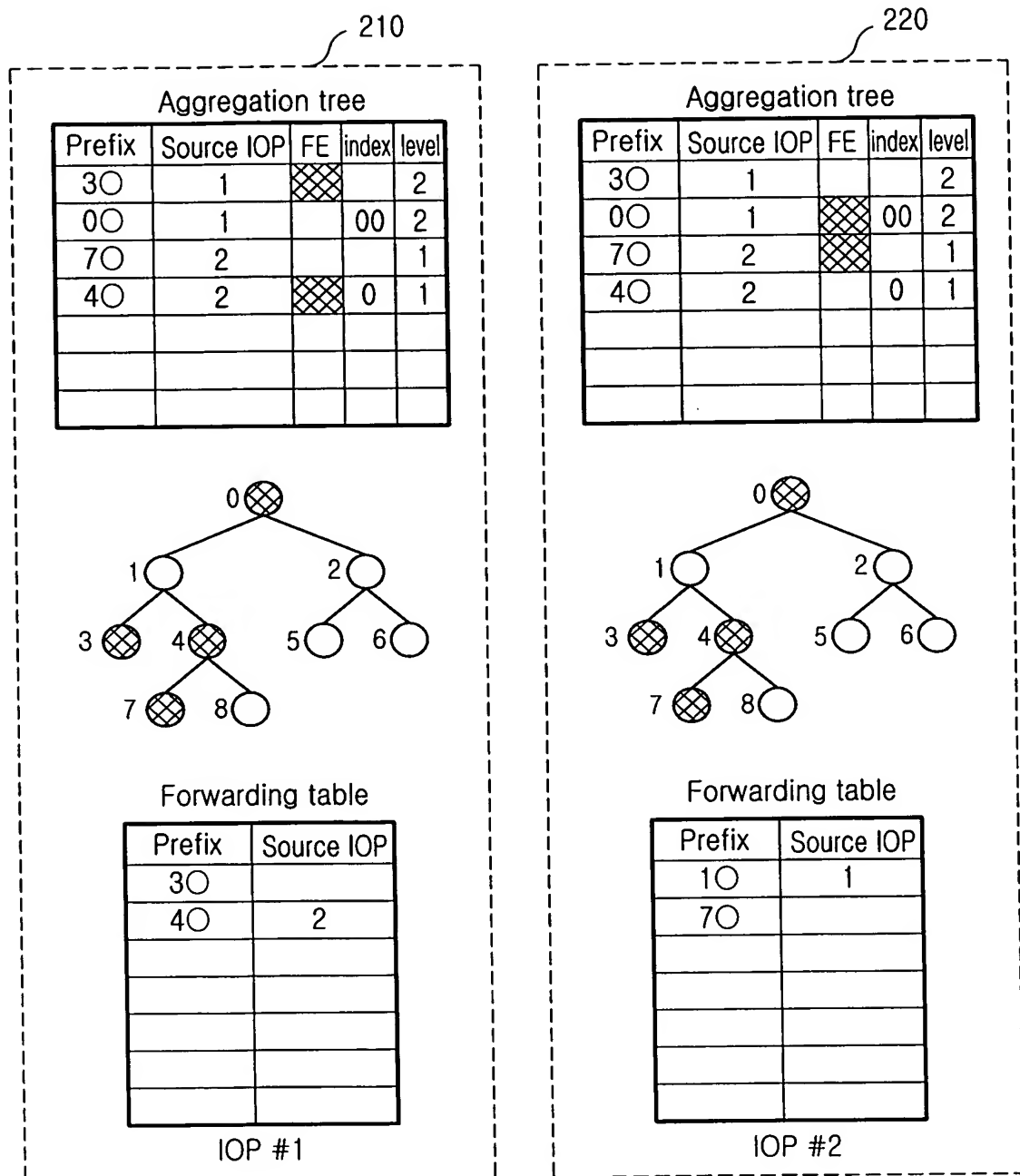


FIG.9A



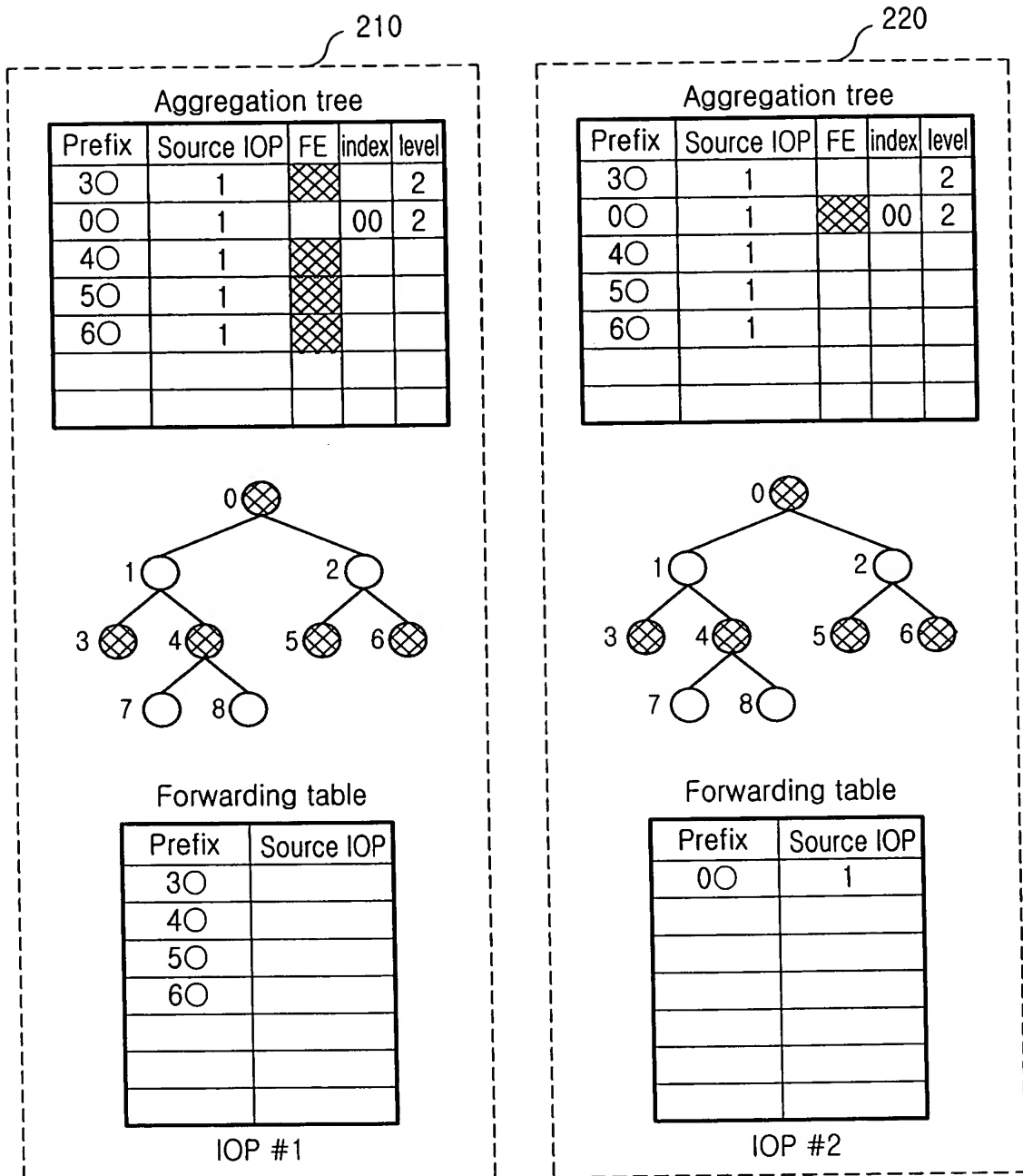


FIG.10A

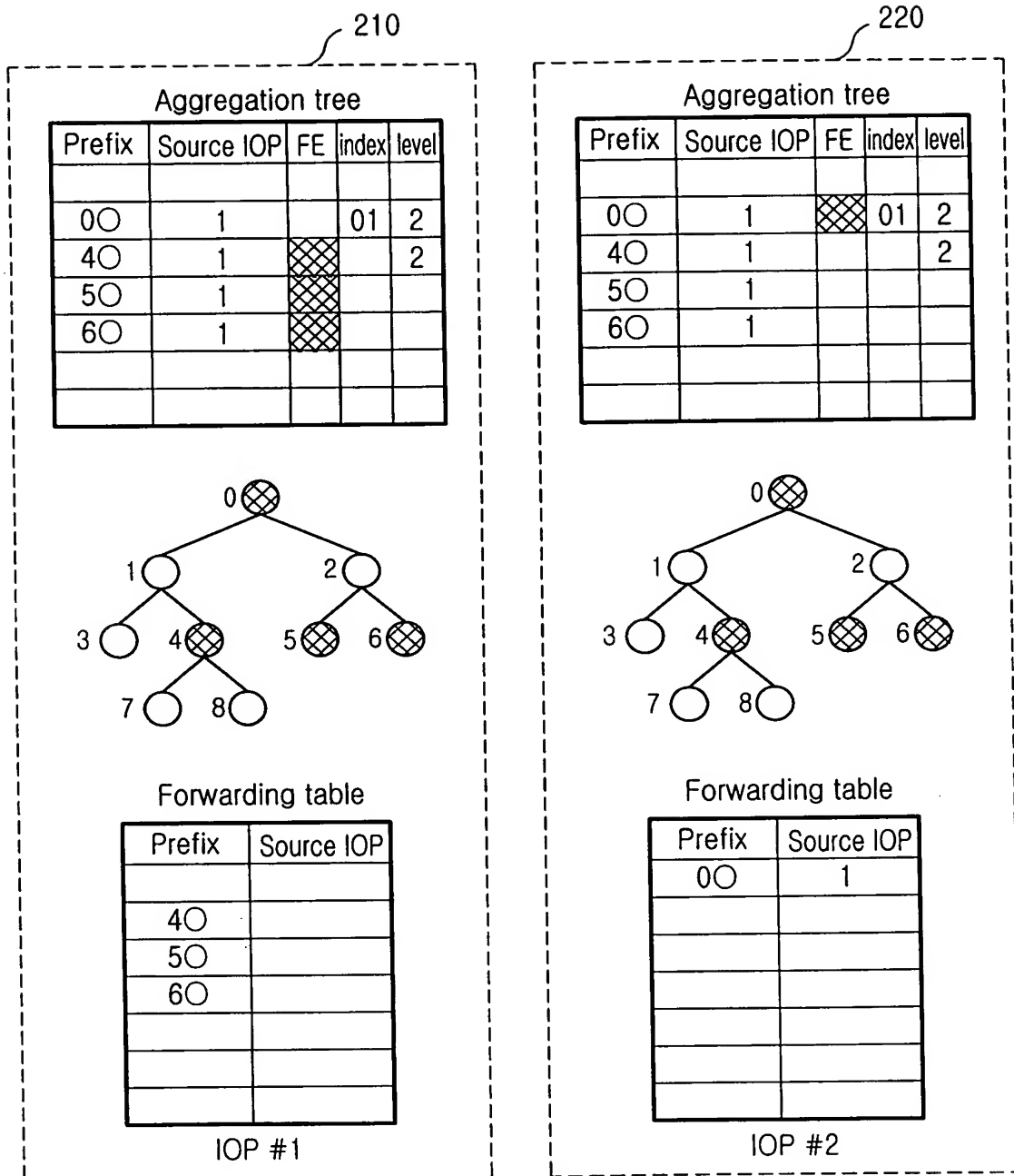


FIG.10B

Default level	Number of the entries	Number of the difference	Reduction rate
1	39,802	58	26%
2	30,574	104	43%
3	25,231	87	53%
4	20,491	118	62%
5	16,478	82	69%
6	12,554	61	77%
Original	53,632	0	0%

FIG.11A

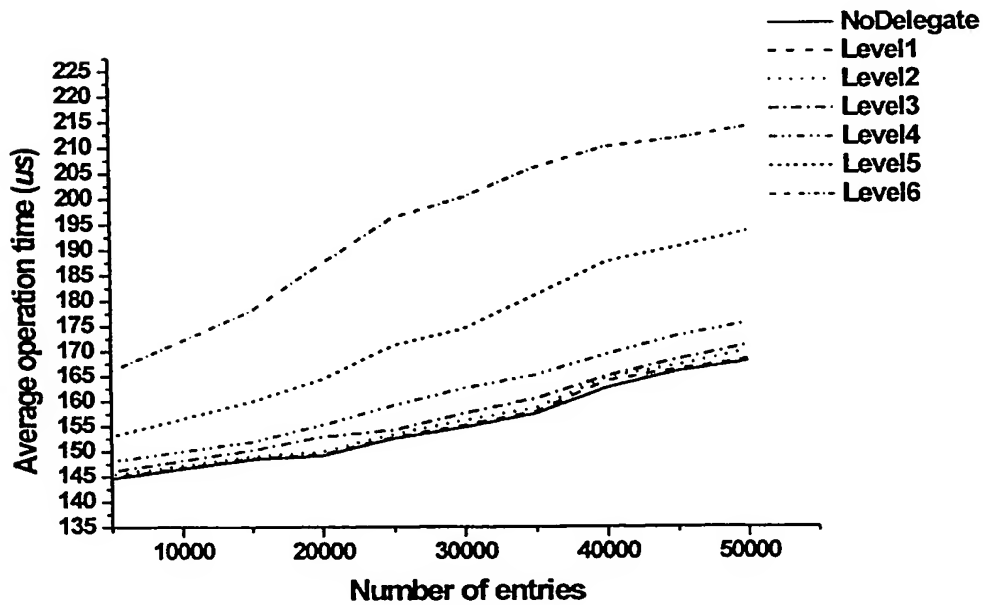


FIG.11B

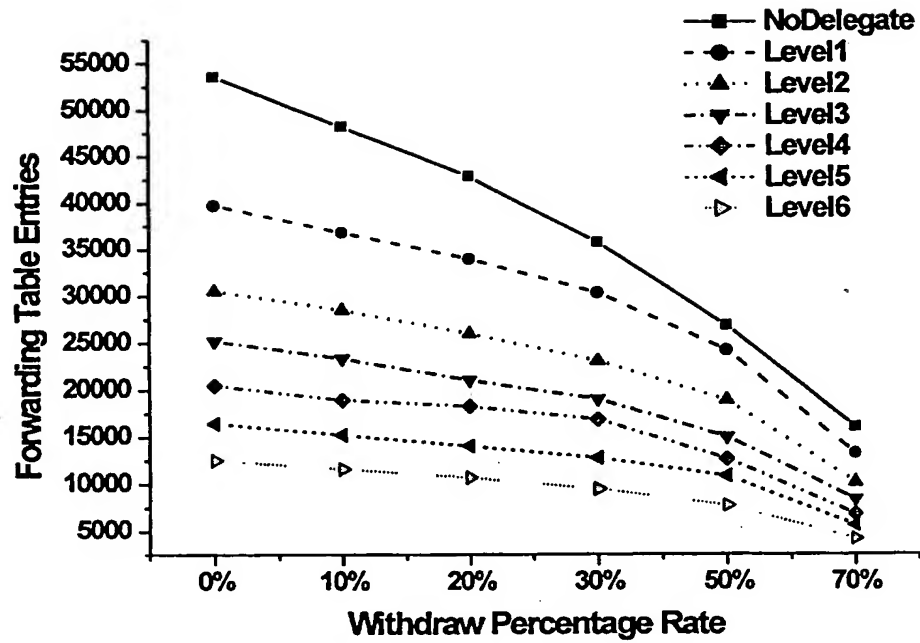


FIG.11C

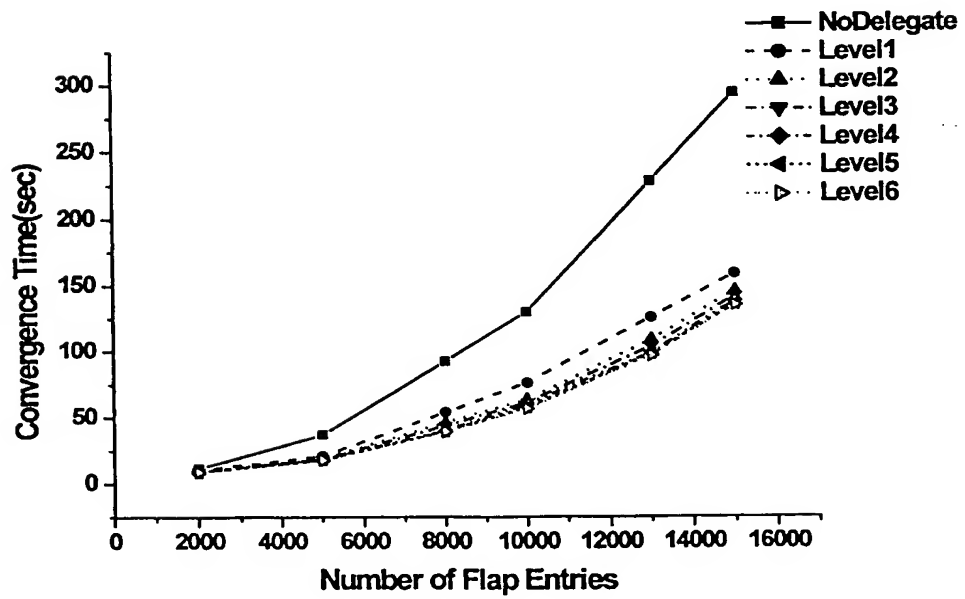


FIG.11D